CS 254 Final Project Report

Ben Crystal and Andrew Hollar

December 13, 2018

Contents

1 Introduction

The problem we are addressing is that we do not have access to a machine that can categorize the origin or environment of a distinct sound. This is an important first step into being able to classify any sound given its data in a file. This could eventually be used for a number of purposes. One application of a future iteration of our algorithm would be being able to identify what environment some AI is in. At the moment, we are focusing on identifying 50 distinct sounds, but in a future iteration of this project, the algorithm would be able to identify the origin of any audio sample. It would make its prediction by identifying and comparing all of the unique sounds within that sample to those in its learned data. An example of an application of this algorithm could be placing an AI robot in any environment and, by recognizing its environment through sound, having it understand how it should interact with its surroundings. For example, a chef robot could identify a kettle or pot boiling by its noise or a timer dinging to recognize its next task. Another potential application of this method would be to allow vocal input for smart home systems (e.g. different audio inputs will trigger specific functions throughout the house to activate). Additionally, one could synchronize this sound classifier with UI in up-coming smart glasses technology, which would allow deaf users to have a visual indication of detected surrounding sounds (e.g. a car on their left, a dog behind them, etc).

The model that we have employed to analyze our dataset is a convolution neural network (CNN). It is recommended to process as large of a dataset as possible, so we used data augmentation to create alternative versions of each sample. These versions include sound files rescaled to different pitches and different time durations without altering the overarching shape of the data. By expanding the size of our dataset, we will improve the robustness of the system.

Using a convolutional neural network on spectrogram images of each sound file, we were able to achieve a testing accuracy of 74.4 percent, which is within 10 percent of what an average human is capable of. We believe that a higher accuracy could be achieved with more parameter tweaking, as well as with further data augmentation.

2 Problem Definition and Algorithm

2.1 Task Definition

The inputs to our algorithm are the audio files from our dataset preprocessed into spectrograms. From those spectrograms, our neural network will extract a variety of features. The output will be a 50x1 vector. From which, the largest number will represent the specific source of the sound (e.g. dog, rooster, pig, etc.).

Furthering this algorithm may help future AI learn how to better engage with its surroundings at any point in time. It may also be helpful in assisting the deaf in better "sensing" their surroundings.

2.2 Algorithm Definition

The baseline algorithm that we utilized to process our data is the convolutional neural network (CNN). CNN algorithms have been proven to be very effective with image classification. A CNN generally uses far less weights than a standard neural network, so it less memory intensive, but more computationally intensive. The architecture of our CNN can be viewed below in Figure 1.

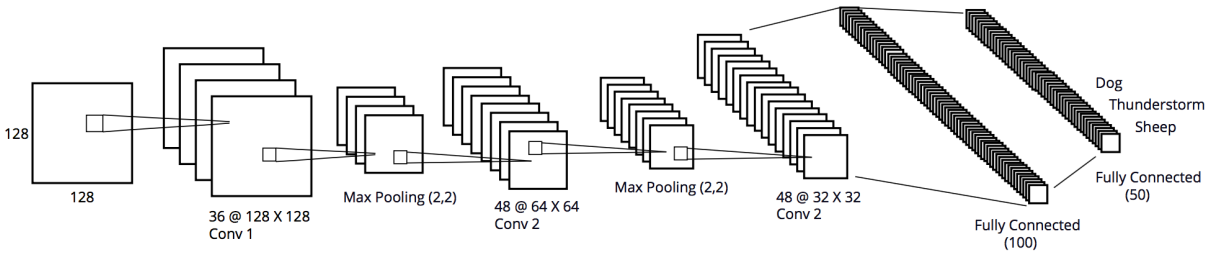

Figure 1: Architecture Diagram of our CNN

Before sending our data into the CNN, we preprocessed the dataset to optimize it for analysis. This process can be seen in Figure 2, and is described in further detail in the "Dataset" subsection. We input the spectrograms to the first convolutional layer. This layer contains 36 filters that are 3x3 pixels in dimension. These filters are slid across the spectrograms in 1x1 pixel strides. These filters are convolved with each consecutive set of pixels in the spectrograms. The outputs of the convolution become the inputs to the max pooling layer. This reduces the size of the feature maps to be 64x64 pixels in dimension. These new feature maps become the input to the Relu activation layer, which cleans up the current data by setting all negative components equal to 0. This acts as the input to the second convolutional layer. The same steps are repeated with 48 filters, followed by max pooling and Relu activation again. This outputs into third convolutional layer of 48 filters, which is then put through another Relu activation layer. Next, the output goes through a flattening process, which converts the current array into a single row vector. This will soon be needed, as the dense layers can only receive single row vectors as input. It is then put through the dropout layer, which randomly ignores 30 percent of the neurons, which helps prevent overfitting as well as reduces the training

time for each epoch. Then, the neurons enter the dense layer, where each input is tied to each output with a relative weight. The neurons are then processed through another Relu activation layer, as well as another 30 percent dropout layer. The last dense layer ties each neuron to each of the fifty available output classes with the weight representing the relative chance of the sample deriving from each class. Finally, the calculated likelihoods assigned to each class are normalized through the Softmax activation layer.

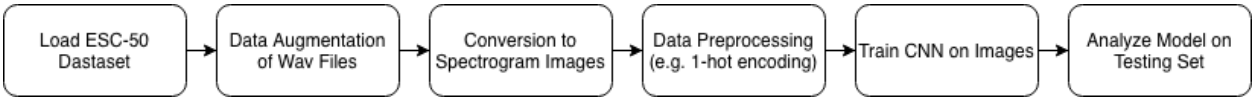

Figure 2: Processing Diagram of our Project

2.3 Dataset

The dataset used for this project is freely available from github. Our dataset contains 40 five-second long recordings of 50 source sounds. These sources are prearranged into five overarching categories: "Animals", "Natural soundscapes and water sounds", "Human, nonspeech sounds", "Interior/domestic sounds", and "Exterior/urban noises". The metadata for each sample includes the particular source. The dataset is labelled with an integer corresponding with the particular source of the sound (e.g. class 0 are "dog sounds"). Since there are 2000 samples and convolutional neural networks function best with larger datasets, we expanded our dataset via data augmentation. We reproduced 3 alternative versions of each audio file– one sped up by 1.07x speed, one slowed down to .81x speed, and one pitch shifted up by 2 half steps. This lead to the total dataset size being 8000 samples. Since the process begins by creating spectrograms of our audio samples, we decided to use Librosa's MEL Spectogram rather than the others available as there were issues in the graphs during moments of pure silence in the recordings that were fixed in Librosa's implementation. Finally, we trimmed all of our spectrograms to 2.97 seconds to ensure that all files, whether sped up or not, would have the same length of data. We did not need any special hardware or GPUs to improve processing speed beyond what our laptops are capable of within a reasonable amount of time.

3 Experimental Evaluation

3.1 Methodology

We are using four metrics to evaluate our algorithm. These include the loss and the accuracy of our model on both the training and testing datasets. The loss is computed based on the categorical cross-entropy. The accuracy is given by the "accuracy" metric within Keras. Our hypothesis for this project was that we will

be able to correctly assign labels to sound files at a success rate similar to that of humans. Our dataset has exactly the same number of samples from each class. This helped us decide that accuracy of correct classification in the test set would be the most important evaluation metric, as opposed to a case like the Cancer dataset example from class that utilized recall as its evaluation metric.

We randomly selected 6000 samples from our augmented dataset (described in the "Dataset" section) to use as our training set, 500 samples for our validation set, and 1500 samples were used as the testing set. We ran a simulation of 100 epochs with a Keras Early-Stopping callback to terminate training when the validation accuracy failed to improve for 4 consecuctive epochs before any significant over-fitting occurred. We recorded the loss and accuracy on both the training and validation datasets at each epoch. Using matplotlib, we plotted the accuracy and the loss against the epoch.

3.2 Results

We were able to achieve a validation accuracy within 10 percent of humans capabilities. Our model performed at 74.4 percent accuracy compared to the human's 81.3 percent, which was found in the Piczak article. After 41 epochs, the loss for the test set was 1.94 and the training set was 0.23, as can be seen in Figure 4. The accuracy of the training set was 95.4 percent and the testing set was 74.4 percent, as can be seen in Figure 3.

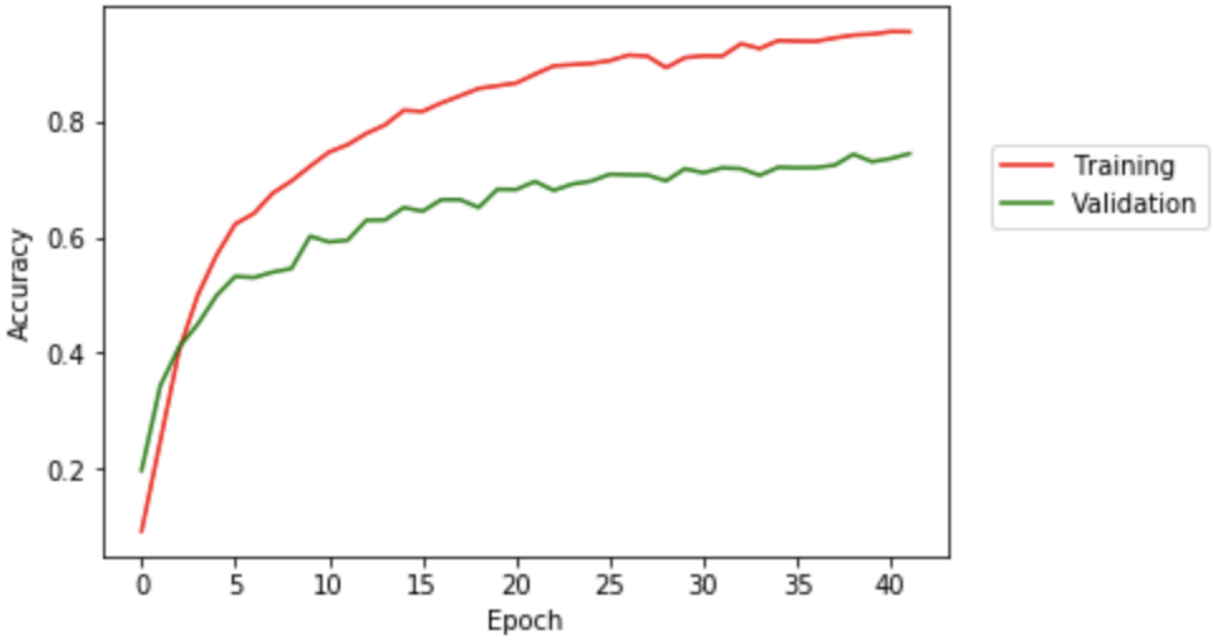

Figure 3: Accuracy over 41 Epochs

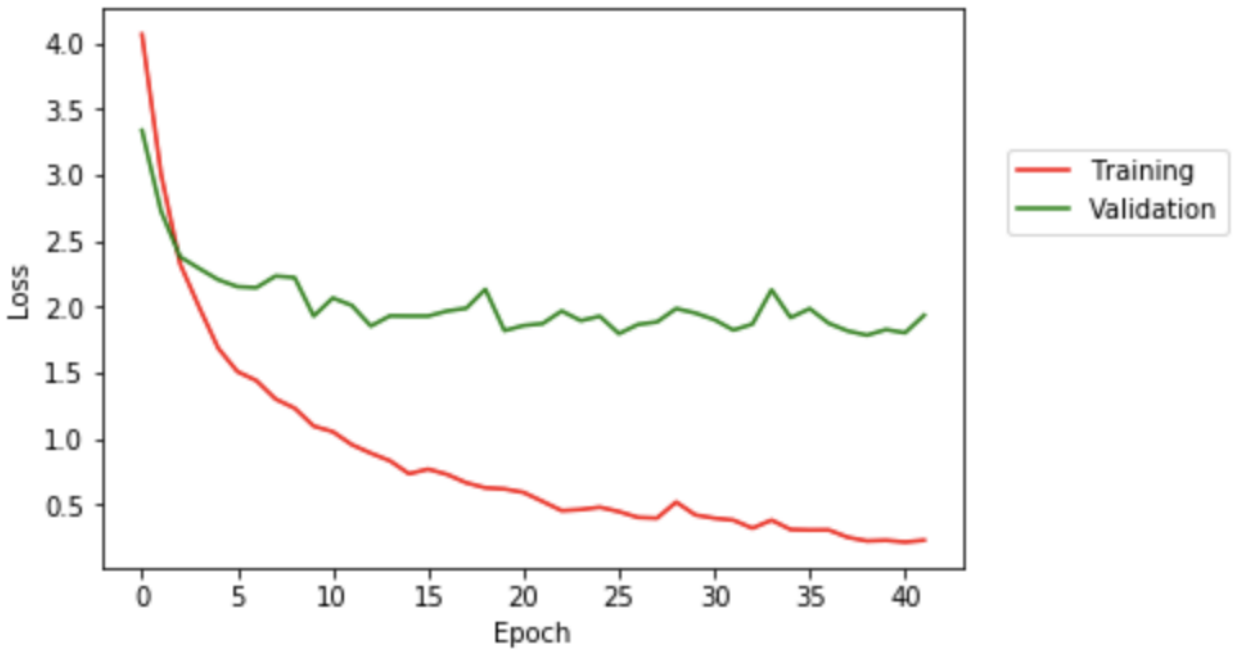

Figure 4: Loss over 41 Epochs

3.3 Discussion

Our hypothesis is supported by our evaluation data. We have achieved a model that can accurately classify the sounds in our dataset at a rate slightly lower than an average human. The Piczak article created an SVM, k-NN, and a Random Forest network as baselines to compare to their CNN. However, they were not able to achieve over 40, 35, and 45 percent accuracy respectively through these methods. The CNN is generally well accepted as the best method for image classification, as it combines the convolution operator (comparing each filter to each image section) with the benefits of a standard neural network, which includes non-linear processing and fault tolerance. Though we would like to be able to maintain a testing accuracy above that of a human, this may be difficult without a larger dataset. To fix the lower accuracy, the dataset could be expanded via further augmentation or having a larger original dataset.

A weakness of our system is that we have to convert the Wav file to a spectrogram before processing the 'sounds' in our CNN. If this technology were to be implemented into, for example, smart glasses technology, there would be overhead in creating the spectrograms for classification through the CNN. Also, the CNNs that we are familiar with can only process images of the same size, which in our case means that all data files needed to be cut to approximately 60 percent of their total length. Essentially, 40 percent of the dataset was lost to be able to process the CNN.

4 Related Work

The first relevant work we found to be the most similar to our particular process. The author of this article created the data set that we will be using. Entitled "ESC: Dataset for Environmental Sound Classification", this article addresses the issue of trying to identify the source of the sound in 250,000 unlabeled auditory excerpts from the "Freesound" project (a website where users upload sound files for music producers to sample). The authors used some of the provided ESC-50 dataset's recordings to train three machine learning algorithms (k-NN, Random Forest, and SVM) to be able to classify the remaining recordings in the dataset. They hoped to apply these algorithms to those samples from "Freesound" as a labelling algorithm. We attempted to use different networks to solve a similar problem of sound classification. We have split our dataset into training, testing, and validation sections. While this research was focused on labeling new sounds from "Freesound", we focused on maximizing the algorithm's ability to differentiate these 50 sounds from within our dataset.

The next relevant work is entitled "Unsupervised word discovery from speech using automatic segmentation into syllable-like units". This group attempted to develop a machine learning algorithm that breaks up speech into syllables and rests, whereby knowing the end of each syllable within the same word would limit the potential number of following syllables (assuming the source of speech was saying recognized statements), shortening the time needed to process each word. These syllable-like features (also known as phonemes) are very helpful in being able to classify what a specific audio source's meaning is. However, in sound origin recognition algorithm, there is an infinite variety of potential sounds. This makes it difficult to break down a sound into its specific syllables or phonemes. Our algorithm will be trained on full sounds, which will allow non-uniform sounds to be generalized into a greater category. One can think of the joke comparing how different dialects of English pronounce words like "potato" and "tomato" differently, but there are only a few specific paths to define these word's potential syllable structures. If we were to look at the onomatopoeia for a non-verbal sound like thunder, for example, there are more potentials than "crackle"– there can also be "kabooms", "rumbles", "buzzes", and hundreds of other distinct sounds in infinite possible combinations and durations.

A third relevant work is entitled "Learning Environmental Sounds with Multi-scale Convolution Neural Networks". This algorithm attempts to improve the frequency resolution and learning filters across all frequencies in environmental detection algorithms based by employing multi-scale convolution operations in their machine learning algorithms. In our project, we attempted to employ a convolutional neural network to classify the sounds in our dataset. Traditionally, CNNs are used in image classification. However, once we

converted our audio files to a spectrogram, we had an "image" of our sound. From here, we were able to use CNN techniques for our image classification. Once we built the CNN, we tried to optimize the parameters to minimize the network's error rate when classifying the sounds.

5 Code and Dataset

The code, as well as comments for reproducing our results, can be found on github. To ensure that the code will run, make sure the dataset and the code itself are in the same directory on your computer. Within this directory, there should be folders called "Models", "PitchChangeWavFiles", "RecordedFromMicrophone", and "TimeStretchWavFiles". Within "PitchChangeWavFiles", there should be a folder entitled "Change1", and within "TimeStretchWavFiles", there should be two subfolders entitled "fast" and "slow".

If anything in the code is not obvious for replication purposes, please contact us at ahollar@uvm.edu and bcrystal@uvm.edu.

6 Conclusion

We were able to achieve a fairly high testing accuracy for identifying sources of sounds from these 50 classes, which is within 10 percent that of humans, as is further addressed in the "Results" section. Although we have attempted to optimize the parameters used in our convolutional neural network for maximum validation and testing accuracy, further improvements could be made. We believe that the largest improvements would be through increasing the size of the training dataset via either more data being recorded or more data augmentation.

7 Bibliography

Karol J. Piczak, October 30, 2015, ESC: Dataset for Environmental Sound Classification, Warsaw University of Technology

Okko Rasanen, Gabriel Doyle, and Michael C. Frank, 2015, Unsupervised word discovery from speech using automatic segmentation into syllable-like units, Aalto University, Department of Signal Processing and Acoustics, Finland, and Stanford University, Language and Cognition Lab, California, United States

Boqing Zhu, Changjian Wang, Feng Liu, Jin Lei, Zengquan Lu, and Yuxing Peng, March 25, 2018, Learning Environment Sounds with Multi-scale Convolution Neural Network, Changsha, China, National University of Defense Technology